Review of Probability

CS 3753 Data Science

Prof. Weining Zhang

Topics

- Events and Probability
- Conditional Probability and Independence
- Beyes Rule
- Discrete Random Variables
- Continuous Random Variables
- Expectation and Variance
- Use SciPy.Stats

SciPy.Stats Package

The SciPy.Stats Package (https://docs.scipy.org/doc/scipy/reference/stats.html) contains a large number of probability distributions as well as a growing library of statistical functions.

- Continuous distributions
- Multivariate distributions
- Discrete distributions
- Statistical functions

In [ ]:
```
from scipy import stats
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%pylab inline
```

Some Functions to Help Plotting

In [ ]:
```
def prob(rv, a, b):
    return 1-(rv.cdf(a)+(1-rv.cdf(b)))

def plotDist(x, func, title, l, xlabel, ylabel) :
    #plt.figure(fig)
    plt.plot(x, func, 'bo', ms=4, label=l) # plot func for elements of x
    xl = plt.gca().get_xlim()
    plt.hlines(0, xl[0], xl[1], linestyles='--', colors='#999999') #lines on Y-axi
s
    plt.gca().set_xlim(xl)
    plt.vlines(x, 0, func, colors='r', lw=2, alpha=0.5) # lines on X-axis
    plt.legend(loc='best', frameon=False)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.show()

def plotDist2(x, func, title, l, xlabel, ylabel) :
    plt.plot(x, func, 'b-', lw=2, alpha=0.6, label=l) # plot func for elements of
x
    xl = plt.gca().get_xlim()
    plt.hlines(0, xl[0], xl[1], linestyles='--', colors='#999999') #lines on Y-axi
s
    plt.gca().set_xlim(xl)
    #plt.vlines(x, 0, func, colors='r', lw=2, alpha=0.5) # lines on X-axis
    plt.legend(loc='best', frameon=False)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)

def plotHistDist(func, x, r, title, l, xlabel, ylabel):
    plt.hist(r, normed=True, histtype='stepfilled', alpha=0.2)
    plotDist2(x, func, title, l, xlabel, ylabel)
```

Experiments and Events

Experiment is to make observations, events are outcomes

- Simple event is non-decomposed (a sample point)
- Compound event is composed of simple events

Probability

- A Sample Space (or Population) S is the set of all (can be infinitely many) simple events.
- Probability $P(A)$ is a number assigned to event A
  - $0 \leq P(A) \leq 1$
  - $P(S) = 1$
  - $P(A_1 \cup A_2) = P(A_1) + P(A_2) - P(A_1 \cap A_2)$ for any $A_1, A_2 \subset S$

Example

A box contains 500 colored balls, with n_1 in red, n_2 in green, n_3 in white, n_4 in blue and n_5 in black. If a ball is selected randomly from the box, what is the probability that

- a red ball is selected?
- a red or a white ball is selected?
- Neither a black nor a blue ball is selected?

Let A be the event that a red or a black ball is selected and B be the event that a red or a white ball is selected.

- What is $P(A \cup B)$, i.e., the probability that A or B occurs?

In [ ]:
```
colors = ['white', 'red', 'green', 'blue', 'black']
balls = pd.DataFrame({'number': [50, 100, 145, 120, 85]}, index=colors)
balls.index.name = 'colors'
total_balls = balls.sum()[0]
balls['prob'] = balls.number/total_balls
print(balls, "\n")
print("p(red ball) = {0:4.3f}".format(balls.loc['red', 'prob']))
print("p(red or white ball) = {0:4.3f}".format(
    balls.loc['red', 'prob'] + balls.loc['white', 'prob']))
print("p(not black or blue ball) = {0:4.3f}".format(
    1-(balls.loc['blue', 'prob'] + balls.loc['black', 'prob'])))
probA = balls.loc['red', 'prob'] + balls.loc['black', 'prob']
probB = balls.loc['red', 'prob'] + balls.loc['white', 'prob']
probAB = balls.loc['red', 'prob']
print("P(A union B) = {0:4.3f}".format(probA+probB-probAB))
```

Conditional Probability

Probability of event A given that event B has occurred is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

provided $P(B) > 0$

Independent Events

Two events are independent if

$$P(A|B) = P(A)$$
$$P(B|A) = P(B)$$
$$P(A \cap B) = P(A)P(B)$$

In [ ]:
```
print("P(A) = {0:4.3f}, P(B) = {1:4.3f}".format(probA, probB))
print("P(A|B) = {0:4.3f}".format(probAB / probB))
print("P(B|A) = {0:4.3f}".format(probAB / probA))
print("P(A intersect B) = {0:4.3f}".format(probAB))
```

Assume events $B_1, B_2, \ldots, B_k$ partitions S and $P(B_i) > 0$ for $1 \leq i \leq k$, then

Law of Total Probability

$$P(A) = \sum_{i=1}^{k} P(A|B_i)P(B_i)$$

Bayes Rule

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_{j=1}^{k} P((A|B_j)P(B_j)}$$

- A simple case

$$P(B|A) = \frac{P(A|B)P(B)}{P((A)}$$

In [ ]:
```
probBcA = probAB / probA
probAcB = probAB / probB
print("P(B|A) = {0:4.3f}, P(A|B)P(B)/P(A) = {1:4.3f}".format(
       probBcA, (probAcB*probB)/probA))
```

Random Variable

A random variable is a real valued function on a sample space

- The number of times face “3” is observed when a fair die is tossed 10 times
- The number of auto accidents in the city on any chosen date
- The weight of the next person walks into the store

Each sample point provides a specific value and many sample points may have the same value

Example: Random Variable

Let Y be the weight of a person walks into a store.

- $Y = 175$ is a numeric event that a person walks into the store has a weight of 175 pounds.
- The probability $P(Y = 175)$ is the sum of probability of all individuals walk into the store whose weight is 175 pounds

Discrete Random Variables and Their Probability Distribution

A random variable is discrete if it can assume a finite or a countable infinite number of distinct values

- The number of students miss a class
- The number of messages received on your cell phone

A probability distribution of a discrete random variable Y can be defined by

- a formula
- a table
- a graph

that provides $P(Y = y)$ for all y so that

- $0 \leq P(Y = y) \leq 1$
- $\sum_y P(Y = y) = 1$
- expected value of Y is $\mu = E(Y) = \sum_y yP(Y = y)$
- variance of Y is $\sigma^2 = V(Y) = E((Y - E(Y))^2)$
- standard deviation of Y is $\sigma = \sqrt{V(Y)}$

If the distribution is known, we can easily find probability for any set of values of R.V.

In [ ]:
```
values = np.array([1, 2, 3, 4, 5])
probs = pd.Series([0.1, 0.3, 0.15, 0.25, 0.2], index=values)
eY = (probs*values).sum()
vY = (((values-eY)**2)*probs).sum()
print("probabilities =\n", probs, "\n")
print("E(Y)= {:4.2f}".format(eY))
print("V(Y) = {:4.2f}".format(vY))
print("sdt(Y) = {:4.2f}".format(vY**0.5))
```

Probability Functions of a Random Variable

- Probability Mass Function (pmf), also Probability Density Function (pdf), is a function that defines
$$Prob(Y = y)$$
- Cumulative Distribution Function (cdf) is a function
$$Prob(Y \leq y)$$
defined over $(-\infty, \infty)$
- Percent Point Function (ppf) is a function
$$y = ppf(q)$$
such that $q = Prob(Y \leq y)$. So, ppf is an inverse of cdf

Example

A random variable Y may take value from $1, 2, 3, 4, 5$

In [ ]:
```
values = [1, 2, 3, 4, 5]
probs = [0.05, 0.2,  0.5, 0.2, 0.05]
Y = stats.rv_discrete(values=(values, probs))
print("P(Y<2) = ", (Y.pmf(0)+Y.pmf(1)))
print("E(Y) = ", Y.mean())
print("V(Y) = ", Y.var())
print("std(Y) = ", Y.std())
```

Binomial Distribution

- Binomial experiment: Make n observations. Each outcome is success or fail. The probability of observing a success is p.
- Let Y be the total number of success, which can be a value in $\{0, 1, 2, \dots, n\}$
- A random variable Y has a binomial probability distribution based on n trials with success probability p iff (if and only if) the probability mass function is
$$P(Y = y) = \binom{n}{y} p^y (1-p)^{n-y}$$
where $y = 1, 2, \dots, n$ and $0 \le p \le 1$
  - $\mu = E(Y) = np$
  - $\sigma^2 = V(Y) = np(1-p)$

In [ ]:
```
# Plot binomial distribution
n, p = 100, .8
x = np.arange(0, n)
rv = stats.binom(n, p)

label = "n={}, p={}".format(n,p) # use distribution parameters to label plot
# pmf(): probability mass func
plotDist(x, rv.pmf(x), 'binomial pmf', label, '# of positives', 'probability')
```

Example: Binomial Distribution

A lot of 5000 electrical fuse contains 5% defectives. If a sample of five fuses is tested, what is the probability of observing at least one defective?

Solution

Make $n = 5$ trials. Success means observed a defective. For this large lot, it may be ok to assume $p = 0.05$.

Let Y be number of defectives observed. Y can be assumed to have a binomial distribution with $n = 5$ and $p = 0.05$.

The probability of observing at least one defective is $P(Y > 0) = 1 - P(Y = 0)$.

In [ ]:
```
# P(Y>0)
n, p = 5, .05
Y = stats.binom(n, p)
print("Binomial distribution with n={} and p={:4.3f}".format(n, p))
print("P(Y>0)={:4.3f}".format(1-Y.cdf(0)))
```

Random numbers generated from a normal distribution

In [ ]:
```
# random numbers from a Binomial distribution
n, p = 50, .35
rv = stats.binom(n, p)

r = rv.rvs(size=n) # generate random numbers

x = np.arange(0, n)
label = "n={}, p={}".format(n,p) # use distribution parameters to label plot
plotHistDist(rv.pmf(x), x, r,  'binomial pmf', label, '# of positives', 'probabili
ty')
plt.show()
```

Example: Binomial Distribution (2)

Randomly selected 20 individuals from a large company are polled whether they support a new company policy. If in this sample 6 persons support the policy, what is most likely the proportion of all employees who will support the policy?

Solution

Let Y be number of individuals in sample who supports the policy. Y is likely to have a binomial distribution with $n = 20$ and $p = ?$, so

$$P(Y = 6) = \binom{20}{6} p^6 (1-p)^{14}$$

To find p that maximize this probability, solve for p from

$$\frac{d}{dp}(\ln(p^6(1-p)^{14})) = \frac{6}{p} - \frac{14}{1-p} = 0$$

So,

$$p = \frac{6}{20} = .3$$

Example: Binomial Test

It is known that 30% of all persons afflicted by a certain illness recover. In a new drug test, ten people with the illness were selected at random and received the medication; nine recovered. Is the drug effective?

Solution

Let Y be number of people among the ten who recovered.

Assume Y has binomial distribution with $n = 10$ and some unknown $p = ?$

If the drug is useless, the probability of recover will be $p = 0.3$. Under this probability, the average number of recovery, the probability of nine or more recover are as follows.

In [ ]:
```
n, p = 10, .3
Y = stats.binom(n, p)
# print("Binomial distribution with n={} and p={:4.3f}".format(n, p))
print("E(Y)={:3.2f}".format(Y.mean()))
print("P(Y=9)={:7.6f}".format(Y.pmf(9)))
print("P(Y>=9)={:7.6f}".format(1-Y.cdf(8)))
```

Since this very unlikely event has been observed, the drug must be effective.

Poisson Distribution

A random variable Y has a poisson probability distribution iff the pmf is

$$P(Y = y) = \frac{\lambda^y}{y!}e^{-\lambda}$$

where $y = 0, 1, 2, \ldots$ and $\lambda > 0$

- $\mu = E(Y) = \lambda$ (yes, λ is the mean!)
- $\sigma^2 = V(Y) = \lambda$

Poisson distribution is a binomial distribution when n approaches infinity

$$\lim_{n \to \infty} \binom{n}{y} p^y (1-p)^{n-y} = \frac{\lambda^y}{y!}e^{-\lambda}$$

In [ ]:
```
# plot a poisson distribution
mu = 5
rv = stats.poisson(mu)

x = np.arange(rv.ppf(0.01), # ppf is percentage point function
              rv.ppf(0.99))
label = "mu={}".format(mu)
plotDist(x, rv.pmf(x), 'poisson pmf', label, 'value of rv', 'probability')
```

Example: Poisson Distribution

A police patrol officer may visit a given location $Y = 0, 1, 2, \ldots$ times per half-hour and each location is on average visited once per half-hour. Assuming that Y has a Poisson distribution, what is the probability that a location is not visited in a half-hour period and what is the probability that a location is visited once, twice, at least once?

Solution

Let $\lambda = 1$. We need to find probabilities for $Y = 0$, $Y = 1$ and $Y > 1$

In [ ]:
```
# Poisson distribution
mu = 1
Y = stats.poisson(mu)

# P(Y=0), P(Y=1), P(Y>0)
print("Poisson disttribution with lambda=1")
print("P(Y=0)={:4.3f}, P(Y=1)={:4.3f}, P(Y=2)={:4.3f}, and P(Y>0)={:4.3f}".format(
    Y.pmf(0), Y.pmf(1), Y.pmf(2), 1-Y.cdf(0)))
```

Zipf Distribution

In [ ]:
```
# plot a zipf distribution
n, a = 20, 1.15
rv = stats.zipf(a)
x = np.arange(0, n)
label = "a={}".format(a)
plotDist(x, rv.pmf(x), 'zipf pmf', label, 'value of rv', 'probability')
```

Continuous Random Variable and Their Probability Distribution

For a R.V. Y, distribution is $F(y) = P(Y \leq y)$ for $-\infty < y < \infty$

- $F(-\infty) = 0$
- $F(\infty) = 1$
- If $y_1 < y_2$, then $F(y_1) \leq F(y_2)$

Y is a continuous random variable if $F(y)$ is continuous for $-\infty < y < \infty$

- $f(y) = P(Y = y) = \frac{dF(y)}{dy}$ is probability density function of Y
- $f(y) \geq 0$ for any value of y
- $P(a \leq y \leq b) = \int_a^b f(y)dy$ (area under the curve between a and b)
- $\int_{-\infty}^{\infty} f(y)dy = 1$
- $E(Y) = \int_{-\infty}^{\infty} yf(y)dy$

Gaussian (or Normal) Distribution

A random variable Y has a normal distribution iff for $\sigma > 0$ and $-\infty < \mu < \infty$, the density function is

$$f(y) = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(y-\mu)^2}{2\sigma^2}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

- $P(a \leq y \leq b) = \int_a^b \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} dy$
- $\mu = E(Y)$
- $\sigma^2 = V(Y)$

If Y has a normal distribution with μ and σ^2, then

$$Z = \frac{Y - \mu}{\sigma}$$

has a standard normal distribution with $\mu_Z = 0$ and $\sigma_Z^2 = 1$

In [ ]:
```
# Plot a Gausian distribution
n, mu, sigma = 20, 0, 2.
rv = stats.norm(mu, sigma)

x = np.arange(mu-0.5*n, mu+0.5*n, 0.01) # range around mu
#for y in x :
#    print("f({0:4.2f})={1:10.8f}\n".format(y, rv.pdf(y)))

label = "loc={}, scale={}".format(mu, sigma)
plotDist(x, rv.pdf(x), 'normal pdf', label, 'value of rv', 'probability')
```

In [ ]:
```
# Normal Distribution
n, mu, sigma = 20, 0, 2.0
rv = stats.norm(mu, sigma)
print("Normal distribution with mu={} and sigma2=()".format(mu, sigma))
a=rv.ppf(0.05)
b=rv.ppf(0.95)
print("P({:4.3f}<=Y<={:4.3f})={:4.3f}".format(a, b, prob(rv, a, b)))
print("P(-inf<Y<inf)={:4.3f}".format(prob(rv, float('-inf'), float('inf'))))
```

In [ ]:
```
# Random numbers from a Normal distribution
n, mu, sigma = 20, 0, 2.0
# Randdom Variable
rv = stats.norm(mu, sigma)
# range on X-axis
x = np.linspace(rv.ppf(0.001), rv.ppf(0.999), 50)
# a set of random numbers from the distribution
r = rv.rvs(size=1000)
# plot histogram of random numbers and overlay the distribution
label = "loc={}, scale={}".format(mu, sigma)
plotHistDist(rv.pdf(x), x, r, 'normal pdf', label, 'value of rv', 'probability')
plt.show()
```

In [ ]:
```
# Normal distribution of the Z-score
n, mu, sigma = 20, 5, 4.0
rv = stats.norm(mu, sigma)

r = rv.rvs(size=1000)
z = stats.zscore(r)
for i in range(1000):
    print(r[i] , ", ", z[i])
plt.hist(r, normed=True, histtype='stepfilled', alpha=0.7)
plt.hist(z, normed=True, histtype='stepfilled', alpha=0.3)
plt.show()
```

Other Distributions

- Exponential Distribution
- Student's T-Distribution
- Chi-Square Distribution

In [ ]:
```
# Plot an exponential distribution
loc, scale = 6, 1  # scale will change height and range
rv = stats.expon(loc, scale)
x = np.linspace(rv.ppf(0.01),
                rv.ppf(0.99), 200)
label = "loc={}, scale={}".format(loc, scale)
plotDist(x, rv.pdf(x), 'Exponential pdf', label,
         'value of rv', 'probability')
```

In [ ]:
```
# Plot a Student's t distribution
df, loc, scale = 4.2, 7, 10
rv = stats.t(df, loc, scale)
x = np.linspace(rv.ppf(0.01),
                rv.ppf(0.99), 200)
label = "df={} \n loc={}, scale={}".format(df, loc, scale)
plotDist(x, rv.pdf(x), 'Student\'s t pdf', label, 'value of rv', 'probability')
```

In [ ]:
```
# Random numbers from a Student's t distribution
x = np.linspace(rv.ppf(0.001),
                rv.ppf(0.999), 200)
r = rv.rvs(size=1000)
label = "df={} \n loc={}, scale={}".format(df, loc, scale)
plotHistDist(rv.pdf(x), x, r, 'Student\'s t pdf', label, 'value of rv', 'probabili
ty')
plt.show()
```

In [ ]:
```
# Plot a chi-square distribution
df, loc, scale = 10, 3, 10
rv = stats.chi2(df, loc, scale)
x = np.linspace(rv.ppf(0.01),
                rv.ppf(0.99), 200)
label = "df={} \n loc={}, scale={}".format(df, loc, scale)
plotDist(x, rv.pdf(x), 'Chi-Square pdf', label, 'value of rv', 'probability')
```

Bivariate Distribution

Y_1 and Y_2 are jointly random variables with a joint distribution function

$$F(y_1, y_2) = P(Y_1 \le y_1, Y_2 \le y_2), -\infty < y_{1,} y_2 < \infty$$

In case of discrete variables

$$F(y_1, y_2) = \sum_{t_1 \le y_1} \sum_{t_2 \le y_2} p(t_1, t_2)$$

where $p(y_1, y_2) = P(Y_1 = y_1, Y_2 = y_2) \ge 0$ for all y_1, y_2, are joint probabilities

In case of continuous variables

$$F(y_1, y_2) = \int_{-\infty}^{y_1} \int_{-\infty}^{y_2} f(t_1, t_2) dt_1 dt_2$$

where $f(y_1, y_2) \ge 0$ for all y_1, y_2, is the Joint density function

- $F(-\infty, \infty) = 1$
- $F(-\infty, -\infty) = F(-\infty, y_2) = F(y_1, -\infty) = 0$
- $P(a_1 \le Y_1 \le b_1, a_2 \le Y_2 \le b_2) = \sum_{a_1 \le t_1 \le b_1} \sum_{a_2 \le t_2 \le b_2} p(t_1, t_2)$ or
- $P(a_1 \le Y_1 \le b_1, a_2 \le Y_2 \le b_2) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} f(t_1, t_2) dt_1 dt_2$

Bivariate Normal Distribution

The multivariate Normal distribution of an n-dimensional vector $x = (x_1, x_2, \dots, x_n)$ may be written

$$p(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu))$$

where μ is the n-dimensional mean vector and Σ is the $n \times n$ covariance matrix.

In [ ]:

```
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

# Our 2-dimensional distribution will be over variables X and Y
N = 60
X = np.linspace(-3, 3, N)
Y = np.linspace(-3, 4, N)
X, Y = np.meshgrid(X, Y)

# Mean vector and covariance matrix
mu = np.array([0., 1.])
Sigma = np.array([[ 1. , -0.5], [-0.5,  1.5]])

# Pack X and Y into a single 3-dimensional array
pos = np.empty(X.shape + (2,))
pos[:, :, 0] = X
pos[:, :, 1] = Y

def multivariate_gaussian(pos, mu, Sigma):
    """Return the multivariate Gaussian distribution on array pos.

    pos is an array constructed by packing the meshed arrays of variables
    x_1, x_2, x_3, ..., x_k into its _last_ dimension.

    """

    n = mu.shape[0]
    Sigma_det = np.linalg.det(Sigma)
    Sigma_inv = np.linalg.inv(Sigma)
    N = np.sqrt((2*np.pi)**n * Sigma_det)
    # This einsum call calculates (x-mu)T.Sigma-1.(x-mu) in a vectorized
    # way across all the input variables.
    fac = np.einsum('...k,kl,...l->...', pos-mu, Sigma_inv, pos-mu)

    return np.exp(-fac / 2) / N

# The distribution on the variables X, Y packed into pos.
Z = multivariate_gaussian(pos, mu, Sigma)

# Create a surface plot and projected filled contour plot under it.
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_surface(X, Y, Z, rstride=3, cstride=3, linewidth=1, antialiased=True,
                cmap=cm.viridis)

cset = ax.contourf(X, Y, Z, zdir='z', offset=-0.15, cmap=cm.viridis)

# Adjust the limits, ticks and view angle
ax.set_zlim(-0.15,0.2)
ax.set_zticks(np.linspace(0,0.2,5))
ax.view_init(27, -21)

plt.show()
```

Marginal and Conditional Probability Distributions

Let Y_1 and Y_2 be jointly random variables.

- Marginal probability

$$P_1(Y_1 = y_1) = \sum_{y_2} P(Y_1 = y_1, Y_2 = y_2)$$

$$P_2(Y_2 = y_2) = \sum_{y_1} P(Y_1 = y_1, Y_2 = y_2)$$

- Conditional distribution of $Y_1 \le y_1$ given that $Y_2 = y_2$

$$P(Y_1 \le y_1 | Y_2 = y_2) = \sum_{t_1 \le y_1} \frac{P(Y_1 = t_1, Y_2 = y_2)}{P_2(Y_2 = y_2)}$$

- Marginal density:

$$f_1(y_1) = \int_{-\infty}^{\infty} f(y_1, y_2) dy_2$$

$$f_2(y_2) = \int_{-\infty}^{\infty} f(y_1, y_2) dy_1$$

- Conditional distribution of $Y_1 \le y_1$ given that $Y_2 = y_2$

$$P(Y_1 \le y_1 | Y_2 = y_2) = \int_{-\infty}^{y_1} \frac{f(t_1, y_2)}{f_2(y_2)} dt_1$$

Example: Marginal and Conditional Probability

From three Republicans, two Democrats, and one independent, a committee of two people is to be randomly selected. Let Y_1 is the number of Republicans and Y_2 be the number of Democrats on the committee. The joint distribution is given in the following table.

- Joint probability distribution: $P(Y_1 = 1, Y_2 = 0) = \frac{3 \cdot 1 \cdot 1}{15} = 0.2$
  - there are three ways to get one Republican, one way to get zero Domestic, and one way to get one independent into the committee
- Marginal distribution of $Y_1 : P_1(Y_1 = 0) = 0 + 0.1333 + 0.0666 = 0.2$

Find the conditional probability that one Republican is selected given that one Democrat is already on the committee.

In [ ]:
```
dist = pd.DataFrame([[0, 0.2, 0.2], [0.1333, 0.4, 0], [0.0666, 0, 0]],
                    index=[0, 1, 2], columns=[0, 1, 2])
dist.index.name = 'Y_2='
dist.columns.name = 'Y_1='
print("The Joint Distribution =\n", dist)
print("marginal probability P(Y_1 = 1) = ", dist.sum()[1])
print("marginal probability P(Y_2 = 2) = ", dist.sum(1)[2])
print("P(Y_1=1 | Y_2=1) = ", dist.loc[1, 1]/dist.sum(1)[1])
```

Independent Random Variables

- If Y_1 and Y_2 are discrete random variables with joint probability $P(Y_1 = y_1, Y_2 = y_2)$, and marginal probabilities $P_1(Y_1 = y_1)$ and $P_2(Y_2 = y_2)$, then Y_1 and Y_2 are independent iff
$$P(Y_1 = y_1, Y_2 = y_2) = P_1(Y_1 = y_1)P_2(Y_2 = y_2)$$

- If Y_1 and Y_2 are continuous random variables with joint probability density $f(y_1, y_2)$, and marginal probability densities $f_1(y_1)$ and $f_2(y_2)$, then Y_1 and Y_2 are independent iff
$$f(y_1, y_2) = f_1(y_1)f_2(y_2)$$

In [ ]:
```
counts = pd.DataFrame({1: [1000, 1500, 2000],
                       2: [2500, 900, 1100],
                       3: [1000, 1000, 850]}, index =[1, 2, 3])
counts.index.name = 'Y_1'
counts.columns.name = 'Y_2'
total = counts.sum().sum()
probs = counts / total
print("counts of pairs of values =\n", counts, "\n")
print("Joint probabilities =\n", probs, "\n")
print("P(Y_1=1, Y_2=2)={0:4.3f}".format(probs.loc[1, 2]))
print("P_1(Y_1=1)P_2(Y_2=2) = {0:4.3f}".format(probs.sum(1)[1]*probs.sum()[2]))
```

Covariance of Two Random Variables

- If Y_1 and Y_2 are two random variables with means μ_1 and μ_2, respectively, the covariance of Y_1 and Y_2 is given by
$$Cov(Y_1, Y_2) = E((Y_1 - \mu_1)(Y_2 - \mu_2)) = E(Y_1 Y_2) - E(Y_1)E(Y_2)$$
- In discrete case:
$$E(Y_1 Y_2) = \sum_{y_1} \sum_{y_2} y_1 y_2 P(Y_1 = y_1, Y_2 = y_2)$$
  and
$$E(Y_1) = \sum_{y_1} y_1 P_1(Y_1 = y_1)$$

- In continuous case:
$$E(Y_1 Y_2) = \int_{y_1} \int_{y_2} y_1 y_2 f(y_1, y_2) dy_1 dy_2$$
  and
$$E(Y_1) = \int_{y_1} y_1 f_1(y_1) dy_1$$
- If Y_1 and Y_2 are independent random variables, then $Cov(Y_1, Y_2) = 0$

In [ ]:
```
vals = np.array([1, 2, 3])
mu_1 = (probs.sum(1) * vals).sum()
mu_2 = (probs.sum() * vals).sum()
mu_12 = (probs * np.array([vals, vals*2, vals*3])).sum().sum()
cov_12 = mu_12 - mu_1*mu_2
print("cov(Y_1, Y_2) = {0:4.3f}".format(cov_12))
```

Conditional Expectations

If Y_1 and Y_2 are two random variables, the conditional expectation of a function $g(Y_1)$ given that $Y_2 = y2$ is

- Discrete case:
$$E(g(Y_1)|Y_2 = y_2) = \sum_{y_1} g(y_1)P(Y_1 = y_1|Y_2 = y_2)$$
- Continuous case:
$$E(g(Y_1)|Y_2 = y_2) = \int_{-\infty}^{\infty} g(y_1)f(y_1|y_2)dy_1$$

In [ ]:
```
mu_1_c_Y_2_2=((probs.loc[:, 2]/(probs.loc[:, 2].sum()))*vals).sum()
print("E(Y_1 | Y_2 = 2) = {0:4.3f}".format(mu_1_c_Y_2_2))
```